IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 March 2023, accepted 22 March 2023, date of publication 29 March 2023, date of current version 7 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3262991

RESEARCH ARTICLE

Effects of Sim2Real Image Translation via DCLGAN on Lane Keeping Assist System in CARLA Simulator

JINU PAHK[1], JUNGSEOK SHIM[1], MINHYEOK BAEK[1], YONGSEOB LIM[2], (Member, IEEE), AND GYEUNGHO CHOI[3]

[1]Daegu Gyeongbuk Institute of Science and Technology, Daegu 42988, South Korea
[2]Department of Robotics Engineering, Daegu Gyeongbuk Institute of Science and Technology, Daegu 42988, South Korea
[3]Department of Interdisciplinary Engineering, Daegu Gyeongbuk Institute of Science and Technology, Daegu 42988, South Korea

Corresponding authors: Yongseob Lim (yslim73@dgist.ac.kr) and Gyeungho Choi (ghchoi@dgist.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant 22-DPIC-17 and Grant 22-DPIC-19.

(Jinu Pahk and Jungseok Shim are co-first authors.)

ABSTRACT Autonomous vehicle (AV) simulation using a virtual environment has the advantage of being able to test algorithms in various scenarios with reduced resources. However, there may exist a visual gap between the virtual environment and the real-world. In this paper, in order to mitigate this gap, we trained Dual Contrastive Learning Generative Adversarial Networks (DCLGAN) to realistically convert the image of the CARLA simulator and then evaluated the effect of the Sim2Real conversion focusing on the lane keeping assist system (LKAS). Moreover, in order to avoid the case where the lane is translated distortedly by DCLGAN, we found the optimal training hyperparameters using feature similarity (FSIM). After training, we built a system that connected the CARLA simulator with DCLGAN and AV in real-time. As for the result, we collected data and analyzed them using the following four methods. First, image reality was measured with Fréchet Inception Distance (FID), which we quantitatively verified to reflect the lane characteristics. The CARLA images that passed through DCLGAN had smaller FID values than the original images. Second, lane segmentation accuracy through ENet-SAD was improved by DCLGAN. Third, in the curved route, the case of using DCLGAN drove closer to the center of the lane and had a high success rate. Lastly, in the straight route, DCLGAN improved lane restoring ability after deviating from the center of the lane as much as in reality. Consequently, it convinced that the proposed method could be applicable to mitigate the gap of simulation toward real-world.

INDEX TERMS Intelligent vehicles, vehicle driving, autonomous vehicles, lane keeping assist systems, lane detection, GAN, DCLGAN, FID, autonomous vehicle simulation, CARLA, software-in-the-loop.

I. INTRODUCTION

The autonomous driving industry has largely been expanding at a rapid pace. Autonomous driving has been proposed as a solution to traffic accidents that cause 1.25 million deaths worldwide every year [1]. However, in the United States alone, a total of 392 traffic accidents recently occurred even in one year due to Level 2 autonomous driving, and these reports utterly raised the need for securing the safety of autonomous driving technology [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Razi Iqbal.

Since autonomous driving technology has been required to have diverse validation processes, autonomous driving simulation is drawing attention as a new testbed. Unlike reality, the simulation can implement complex situations, such as traffic accidents and specific weather, that are hard to reproduce in the real-world [3], [4], [5]. This technology gives abundant training datasets about abnormal situations and can potentially prevent traffic accidents more perfectly. Therefore, many studies have been conducted for modeling vehicles in simulation [6], [7], [8].

In order to adapt proven algorithms from virtual-world to real-world, reducing the gap between simulation and reality

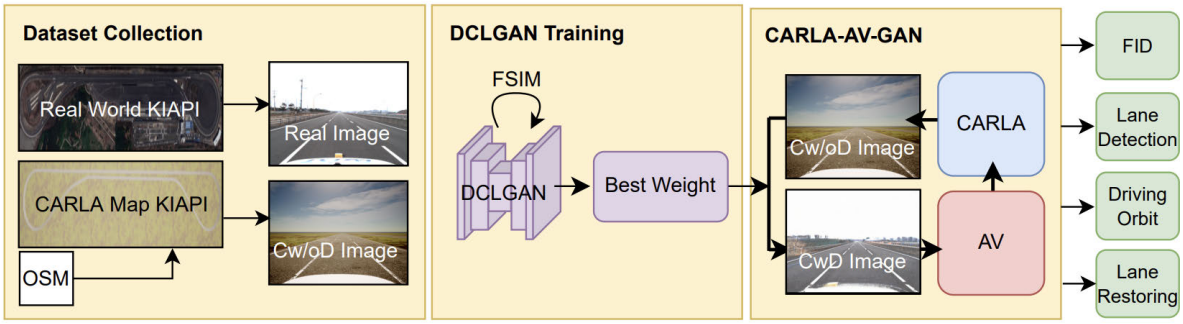


FIGURE 1. Full framework for identifying Sim2Real image gaps and driving gaps from the CARLA-AV-GAN loop.

is an inevitable task. This gap can be caused by several factors such as the unreality of sensor models (e.g., camera, LiDAR, GPS, and IMU), mismatch of dynamic models, and measurement noise in the real environment. Among the factors, the camera sensor input images are the main cause of the gap because this sensor accounts for the largest portion of driving. The image gap which is a difference in visual realism tends to make driving results unreliable. Besides, mitigating this gap only by adjusting the simulation parameters can be almost impossible. Therefore, simulation developers and users often have used a high-level graphic engine to make the visual contents of the simulation look more realistic.

To make images from the two domains similar, Generative Adversarial Networks (GAN) [9] have been adopted. With the development of the GAN structure, the model has developed to be applied to various fields such as colorizing gray image [10] and restoring image quality [11]. Several studies have also been proposed to make the images of the virtual world close to reality [12], [13]. Although the study on improving the graphic quality of the virtual world is active, autonomous driving tests with these photorealistic techniques have not yet been well investigated so far.

The purpose of this paper is to show the possibility of applying GAN to the autonomous vehicle (AV) simulation not just translating images, but also testing driving results by using CARLA [14] simulator. As for the GAN, Dual Contrastive Learning Generative Adversarial Networks (DCLGAN) [15] was selected because DCLGAN was trained better than other unpaired image translation models such as CycleGAN [16] without overfitting even with our less diverse data. Nevertheless, to avoid the lane distortion that DCLGAN sometimes exhibits, we explored how to train in optimal conditions using feature similarity (FSIM).

For this purpose, the data were collected from the three domains: real-world, CARLA with DCLGAN (hereafter referred to as CwD), and CARLA without DCLGAN (hereafter referred to as Cw/oD). The real-world data were obtained from the Korea Intelligent Automotive Parts Promotion Institute (KIAPI) and the virtual-world data were obtained from a virtual map created identically to KIAPI. The types of data were camera images and GPS data. To show the DCLGAN contributed to making more realistic driving results, a method to quantify the Sim2Real gap is needed. There are several original contributions of this paper as following.

First of all, Fréchet Inception distance (FID) [17], which calculates differences in density of two distributions of image sets in the diverse environment, was used to measure the realism of the images of the CARLA simulator. After determining whether FID was suitable for measuring this image gap, we measured the FID of the images converted by DCLGAN. These converted images were also used to measure the accuracy of the lane detection model, an important part of LKAS. We compared CwD, Cw/oD, and real-world lane segmentation accuracy. Furthermore, we compared how much the GPS trajectory of the vehicle driving the virtual KIAPI map deviated from the center of the lane with and without DCLGAN. The ability of these systems to recover from lane departure on a straight line was also measured. The last two results have a differentiated meaning in that they are the driving test, not an image task. Consequently, we found that the above processes show that our proposed CwD method is more effective in practical use of autonomous driving simulation tests such as LKAS.

II. RELATED WORKS

CARLA [14] has been considered one of the reliable autonomous driving research simulator. Thus, we used the CARLA simulator for the implementation of a virtual-vehicle and building environments by using the functions followed. First, the provided ROS-bridge enables the connection between the real-vehicle and the virtual-vehicle. Second, CARLA allowed us to spawn various sensors and configure specific parameters such as the camera's FoV, resolution, and gamma. Lastly, target vehicles can be implemented through Unreal Engine 4 (UE4), the game engine of CARLA. We imported a 3D model of Hyundai IONIQ electric vehicle to spawn sensors at the same position as the real-vehicle.

GAN consists of a generator that creates fake images following real images' distribution, and a discriminator that distinguishes between the fake images and the real images [9]. As the two modules compete repeatedly, GAN can generate

more realistic images. However, the GAN model has a limitation that it necessarily requires paired datasets for training. CycleGAN solves the problem by adding the reverse mapping that returns the fake image to the original image. Whereas cycle consistency loss has a drawback that the fake images have low diversity. This limited output problem of low diversity is well known as mode collapse. Contrastive Learning for Unpaired Image-to-Image Translation (CUT) and Dual Contrastive Learning GAN (DCLGAN) tried to deal with this issue through contrastive learning. Especially, Similarity Dual Contrastive Learning GAN (SIMDCL) with similarity loss added to DCLGAN suffers less from the mode collapse by allocating embeddings as much as the number of data domains [15]. Thus, we mainly used SIMDCL, considering the characteristics of our dataset, which has low diversity and a high risk of mode collapse.

FID is an evaluation metric for a generative model such as GAN, text-to-image, etc. The distance between the fake images and the real images is measured by comparing the extracted features from the pre-trained Inception v3 model with the Imagenet dataset [18]. In this paper, the distance is used to measure how realistic the generative model creates images. FID between true samples (T) and generated samples (G) is calculated as equation (1) using the mean of the variance of features (μ) and covariance ($\sum$). Tr stands for the trace of the matrix.

$$FID(T, G) = |\mu_T - \mu_G|^2 + Tr(\sum_T + \sum_G - 2(\sum_T \sum_G)^{\frac{1}{2}}). \tag{1}$$

ENet-SAD [19] has the characteristic of self-learning, which delivers and enhances the representation top-down by distilling the attention map of the lower layer. With this distillation technique, ENet-SAD can be computed faster without additional annotation or inference even with much fewer parameters compared to other CNN-based lane detection models such as SCNN. Since the IoU of the ground truth lane and lane segmentation prediction results were better than the previous models, our LKAS system uses ENet-SAD.

TuSimple [20] introduces the lane segmentation accuracy formula as equation (2). C_{clip} is the number of true points in each clip and S_{clip} is the number of the ground truth points in the clip. Thus, we used this evaluation method to compare accuracy in each domain.

$$Accuracy = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}}. \tag{2}$$

Yurtsever et al. [12] proposed a rendering method that realistically synthesizes the background of CARLA using Conditional GAN and CycleGAN, while leaving objects of interest such as lanes and vehicles intact. In this way, distortion or disappearance of the object can be prevented, but it is not possible to completely translate the appearance of objects to accurately reflect the properties and patterns of each domain. The model presented by Richter et al. [13] converts the entire image without such hallucinations by using the features of G-buffer information such as geometry, materials, lighting, and segmentation created during the game rendering process in the generator model. However, these studies do not confirm how realistically changed images actually affect the perception or control results of autonomous driving algorithms, or do not compare them with actual driving data. For this comparative study, we trained a GAN model with data that directly reflects the difference in characteristics between our simulation and real domain, and also dealt with the hallucination problem.

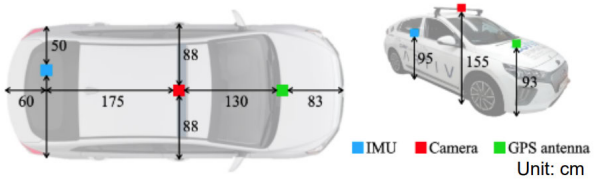


FIGURE 2. Vehicle top view and sensor location (left). Sensor installation on the test vehicle (right).

III. ENVIRONMENT SETTING

A. REAL-WORLD SETTING

1) SPECIFICATION

Table 1 describes the sensors and hardware specifications and Fig. 2 shows the location of the sensors. Although the camera maximum resolution was 1632×1248 pixels, 808×620 was used by binning. The vehicle was the Hyundai IONIQ electric vehicle modified for applying the drive-by-wire system.

2) LKAS ALGORITHM

We have autonomous driving algorithms and architecture that were certified by MOLIT (Ministry of Land, Infrastructure, and Transport) of the Republic of Korea. This certification required autonomous emergency braking (AEB), smart cruise control (SCC), and lane keeping assist system (LKAS). Among the technologies we have, only LKAS function was used in the experiment.

According to previous papers [21], [22], the path of autonomous vehicles was derived through camera, GPS, and IMU sensors. Three paths are obtained through each three sensors. Then, Optimal Path Tracker Selection picks the best path among the paths considering the instability of steering angle, GPS reliability, vision reliability, and road information.

In this study, we only used the path from the camera and designed this algorithm being worked in two stages. The first stage performed in deep learning PC installed on AV is lane segmentation by using the ENet-SAD network model. This ENet-SAD model was trained by datasets from both the CULANE dataset [23] and our dataset. The second stage is lane detection which transforms the segmented image to a bird's eye view through inverse perspective mapping (IPM), and then calculates lane center points by finding the fitting function that best describes the lane among linear, quadratic,

TABLE 1. Sensors and hardware's specification.

Hardware Component	Purpose	Model	Specifications
Main PC	Vehicle Control	Nuvo -8108GC -NX	CPU: Intel Xeon Processor E-2176G Memory: 32GB DDR4 GPU: GeForce RTX-2080Ti 11GB
Deep Learning PC	Running deep learning algorithm	Custom PC	CPU: Intel I7 9700F Memory: 32GB DDR4 GPU: NVIDIA Titan X 12GB
Simulation PC	Playing CARLA Calculating GAN model	Custom PC	CPU: AMD Ryzen 9 3950x Memory: 64GB DDR4 GPU: GeForce RTX-3090 24GB
FLIR Camera	Lane recognition	BFS-PGE -19S4C-C	FoV: 76° FPS: 50 Gamma: 0.8 Resolution: 1632x1248 (808×620 after binning)
GPS (RTK GNSS)	Checking the vehicle trajectory	MRP-2000	RTK Accuracy: 0.010m + 1ppm CEP Rate: up to 10Hz

and cubic least squares fitting [22]. After these two stages, the main PC installed in the AV calculates the steering angle value using the lane detection result and the pure pursuit algorithm, which enters the CARLA vehicle through the ROS topic. In this paper, the lane keeping algorithm was used to obtain data by running a real-vehicle on the KIAPI high-speed main circuit or by running a virtual-vehicle on the virtual KIAPI map.

B. CARLA SIMULATOR SETTING

1) CAMERA AND VEHICLE

CARLA simulator provides various sensors and attributes in the virtual environment. The attributes make it possible to set the sensor specifications. The performance of the camera sensor described in Table 1 was imported such as FoV or Gamma. Then, the camera was spawned according to Fig. 2 position. The CARLA image taken by spawned camera sensor and the real-world image can be also seen in Fig. 3.

2) MAP

To make a custom map, CARLA requires map geometry information in .fbx format and OpenDRIVE information in .xodr format. For these, we used RoadRunner recommended by CARLA as a map editor and OpenStreetMap (OSM) as

FIGURE 3. CARLA image (left) and the real-world image (right).

TABLE 2. Road parameters' specification.

Parameters	Specification
Road Width (m)	3.5
Line Length (m)	4.5
Line Spacing (m)	4
Line Width (m)	0.125
Slope (°)	30 (first lane) 0 (second, third lane)

map data. OSM has latitude and longitude vector data and RoadRunner makes OSM possible to get both map geometry information and OpenDRIVE information.

The process of making the KIAPI map was as follows. First, as shown in Fig. 4(a), we dragged the desired area from the satellite map in OSM and export the file in .osm format. Second, as shown in Fig. 4(b), the latitude and longitude were corrected by comparing the OSM file exported from JOSM with the GPS file of actual measurement data. After that, the calibrated OSM file was imported into RoadRunner, and the RoadRunner file set the road width, lanes, and slope of the road according to the information announced in KIAPI [24]. Fig. 4(c) describes this process and Table 2 shows the specification of each parameter. Lastly, this file was exported in CARLA format such as .fbx and .xodr format files.

3) TIME

The experiment with the same time period as the real-world is not able to be performed because of the calculation time of GAN. As a solution, we used synchronous time. According to CARLA documentation, ''On synchronous mode, the server waits for a client tick, a ''ready to go'' message, before updating to the following simulation step'' [25]. It means that we can set the attribute to wait for a step until the vehicle control command arrives. Therefore, the client server would wait for a cycle described in the next section.

4) NETWORK CONNECTION BETWEEN CARLA AND AV

Three computers were used for the experiments, called the Main PC, Deep Learning PC (DL PC), and Simulation PC. As described in Table 1, Main PC decided on vehicle control such as steering angle value and throttle value. DL PC computed deep learning algorithms such as the LKAS algorithm, and the Simulation PC played CARLA simulation. Table 3 describes the ROS version for each computer. For connecting

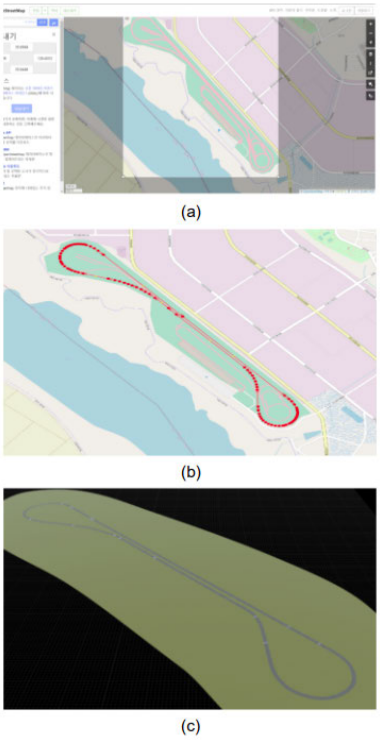


FIGURE 4. The process of creating a virtual map from the OSM data of KIAPI. (a) Exporting map data from OSM. (b) PG OSM file edited on JOSM. (c) KIAPI map produced in RoadRunner.

TABLE 3. Each PC's Ubuntu version and ROS version.

PC Name	Ubuntu Ver.	ROS Ver.
Main PC	18.04 LTS	Melodic
Deep Learning PC	18.04 LTS	Dashing
Simulation PC	20.04 LTS	Foxy

different versions of ROS 2 between DL PC and Simulation PC, Cyclone DDS is applied to Simulation PC and Fast RTPS is used to DL PC.

Fig. 5 shows the flow of the three computers interactions with ROS topic names. The proposed method with this computer configuration was accomplished as following three procedures. First, in the Simulation PC, the camera sensor from CARLA gives an image to DCLGAN. The output from the DCLGAN model is sent to the DL PC. Second, in the DL PC, lane_segmentation function is performed and gives segmentation data to lane_detect function. The lane_detect function finds the center points of the lane and sends the output points to the Main PC. Third, in the Main PC, lane_driving function gets vehicle status from the Simulation PC and center points from DL PC. By utilizing these values, the lane_driving function determines the steer value. The move_car function

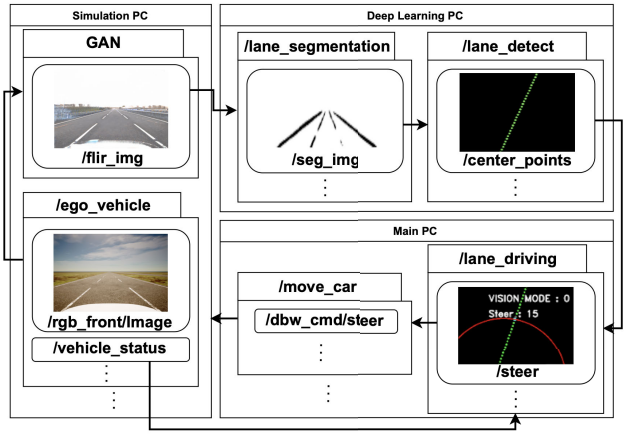


FIGURE 5. CARLA-AV-GAN connection overview.

calculates proper steer angle, throttle, and brake values. Then the function sends them to the Simulation PC for control of the vehicle.

5) TRAINING DCLGAN

We tried to reduce this Sim2Real gap with DCLGAN. The data of the real-world domain was collected from the high-speed circuit of KIAPI and the data of the CARLA domain was collected from the OSM map of KIAPI. Both were collected while driving at a speed of 30 km/h in the second lane. The CARLA domain contains 5498 images as train set and 611 images as test set, and the real-world domain contains 6929 images as train set and 770 images as the test set. In order to match the channels of the FLIR camera images used in the AV, the order of the CARLA image channels was changed from RGB to BGR.

The images used for training the model were loaded with a size of 600×600 and cropped to a size of 300×300. The deep learning network model was selected as SIMDCL and the rest of the options kept the DCLGAN default settings. Additionally, $\lambda_{NCE,X}$, a hyperparameter of the loss function, was determined through the process as described in equations (3)-(5).

As shown in Fig. 6, if the image passes through the DCLGAN model, the lane in CARLA may not maintain its original shape and may be distorted enough to affect lane-keeping driving. This lane distortion occurs when the generator of the GAN does not fully learn the data distribution and is fixed only in a specific biased direction to generate an image [15]. Also, the imbalance between the straight line data and curved line data of the lane can be the cause.

In order to solve the problem of distortion of the lane shape that occurs when training with the default settings of the model, we tried to find the optimal hyperparameter set by adjusting the coefficients of the loss function (Hyperparameter tuning). Equation (3) shows the loss function of DCLGAN, and equation (4) shows the loss function of SimDCL [15]. G is the X domain(in our paper, Carla)→ Y

domain(Real-world) generator, and F is the $Y \rightarrow X$ generator. D_X distinguishes the generated X images from the real images, and D_Y distinguishes the Y images. H_X and H_Y refer to vectors embedded by G encoder and H encoder, respectively. When converting $X \rightarrow Y$ or $Y \rightarrow X$, it uses the normal GAN loss (L_{GAN}) for G or F. In addition, PatchNCE loss ($L_{PatchNCE}$) is used, which learns to maximize the similarity between a patch in the generated real-world image and its positive pair in the original CARLA image while minimizing the similarity between the real-world patch and negative CARLA pairs. Also, Identity loss ($L_{identity}$) is calculated to preserve the color of the existing image, and Similarity loss (L_{sim}) is calculated to measure whether the real image and the generated image have similar styles in the same domain. The default coefficients for each loss term are 1 for *GAN*, 2 for *NCE*, 10 for *sim*, and 1 for *idt*.

As in equation (5), the X part of the PatchNCEloss term is associated with a generator that converts $X \rightarrow Y$. Where $\widehat{z}_l^s$ is the query patch of the generated image, z_l^s is the positive patch of the original image, and $z_l^{S\setminus s}$ is the negative patches. It calculates the entropy-cross loss (l) for all layers and patches. If the query is closer to the positive paired patch and farther from the negative patches, it will try to retain more features of the CARLA patch. We checked whether better results that keep the direction or shape of the lane can be obtained by changing the coefficient such as $\lambda_{NCE,X}$.

$$\begin{aligned} &L_{DCLGAN}(G, F, D_X, D_Y, H_X, H_Y) \\ &\quad = \lambda_{GAN}(L_{GAN}(G, D_Y, X, Y) \\ &\qquad + L_{GAN}(F, D_X, X, Y) \\ &\qquad + \lambda_{NCE} L_{PatchNCE_X}(G, H_X, H_Y, X) \\ &\qquad + \lambda_{NCE} L_{PatchNCE_Y}(F, H_X, H_Y, Y) \\ &\qquad + \lambda_{idt} L_{identity}(G, F). \end{aligned} \tag{3}$$

$$\begin{aligned} &L_{SimDCL}(G, F, D_X, D_Y, H_X, H_Y) \\ &\quad = \lambda_{GAN}(L_{GAN}(G, D_Y, X, Y) \\ &\qquad + L_{GAN}(F, D_X, X, Y) \\ &\qquad + \lambda_{NCE} L_{PatchNCE_X}(G, H_X, H_Y, X) \\ &\qquad + \lambda_{NCE} L_{PatchNCE_Y}(F, H_X, H_Y, Y) \\ &\qquad + \lambda_{sim} L_{sim}(G, F, H_X, H_Y, H_1, H_2, H_3, H_4) \\ &\qquad + \lambda_{idt} L_{identity}(G, F). \end{aligned} \tag{4}$$

$$\begin{aligned} &L_{PatchNCE_X}(G, H_X, H_Y, X) \\ &\quad = E_{x\sim X} \sum_{l=1}^{L} \sum_{s=1}^{S_l} l(\widehat{z}_l^s, z_l^s, z_l^{S\setminus s}). \end{aligned} \tag{5}$$

In this case, FSIM [26] was used to evaluate the degree of lane distortion of the generated image in the case of each adjusted hyperparameter. The edge map similarity between the original image and the fake image is measured from the phase congruency (PC) map extracted based on frequency information and the gradient magnitude (GM) map extracted based on the gradient of pixel values. The PC map is obtained by calculating the sum of vectors (local energy) consisting of amplitude and phase from frequency information and dividing it by sum of the amplitudes, and the GM map can be obtained by convolving the image with a first-order differential edge detection mask. To compare the edge information of only lanes, as shown in Fig. 7, 808×245 sized images excluding the background and car bonnet were cropped from each 808×620 image of the CARLA test imageset, and the FSIM of each pair of the Cw/oD image and the CwD image was measured. A model with proper $\lambda_{NCE,X}$ that maintains the original lane shape or direction well can have a high FSIM value. The weight determined in this way is then used for FID measurement. The relationship and differences between FSIM and FID are shown in Fig. 8. FSIM can directly measure the similarity between two paired images, but since FID compares the distribution of images, it is necessary to compare image sets when measuring FID.

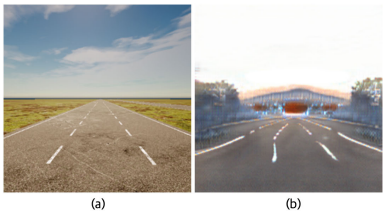


FIGURE 6. This is a mode collapse in which DCLGAN learned only in the direction of cheating the discriminator rather than generating a wide range of outputs. (a) Original image, (b) Distorted lane due to mode collapse.

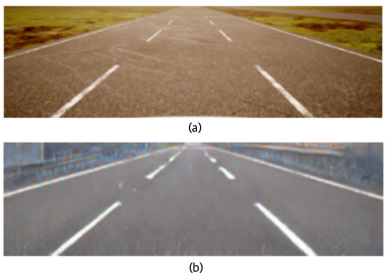


FIGURE 7. (a) A Cw/oD image cropped to 808×245 size (b) A CwD image generated therefrom for FSIM measurement.

Table 4 shows the average FSIM value in each $\lambda_{NCE,X}$. In all cases, the weight that seems to have the least lane distortion among the weights trained more than 15 epochs and less than 60 epochs were selected (because the loss function has largely converged after epoch 15), and the average of the FSIM values before and after translation of 611 images of the CARLA domain test set was measured. For FSIM measurements, quality_metrics and image_similarity_measures Python packages were used [26]. $\lambda_{NCE,X}$ value 3, which has the highest FSIM value 0.451 and is judged to have the least distortion, was determined as the optimal condition.

IV. APPROACH METHOD

A. FID VERIFICATION

It is necessary to guarantee whether the difference in characteristics of lanes can be quantitatively measured.

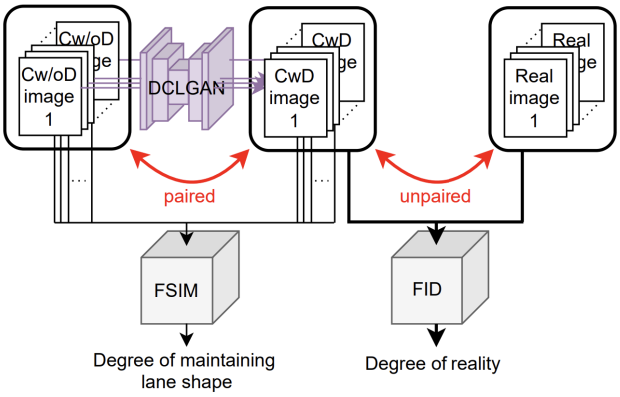


FIGURE 8. A diagram of the relationship between Cw/oD images, CwD images, and real-world images with measuring FSIM and FID from them. When a CwD image is generated from each CARLA image, the FSIM is measured from each pair. FID is measured between unpaired CwD imageset and real-world imageset to measure how realistic the generated images are.

TABLE 4. Mean FSIM values (611 pairs of paired Cw/oD images & CwD images) for each $\lambda_{NCE,X}$.

$\lambda_{NCE,X}$	1	2 (default)	3	4
FSIM	0.439	0.423	0.451	0.403

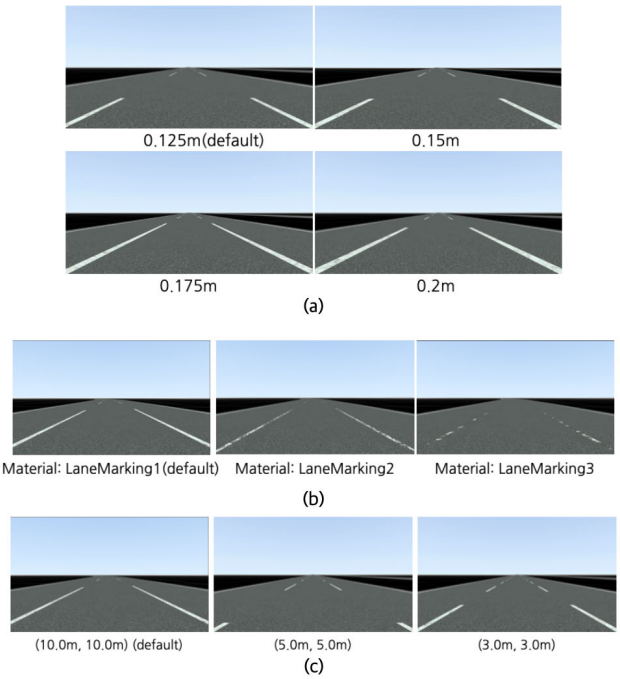


FIGURE 9. Front camera image in RoadRunner Simulation Play according to (a) the thickness of lanes, (b) the material of lane texture, and (c) the length of the lane and the distance between lanes.

As previously shown in Fig. 3, the lanes in the real-world are clearer and thicker than the ones in the virtual environment. To see if the FID used to quantify the Sim2Real difference reflects this well, it is necessary to test the FID by separately creating maps in which the characteristics of these lanes are clearly distinguished. These maps were created based on KIAPI map and varied the thickness, texture, length, and distance of lanes by RoadRunner. Fig. 9 shows these characteristic differences. For the texture, LaneMarking1, 2, and 3, which are lane materials provided by RoadRunner, were used. The material used in LaneMarking1 is the sharpest and 3 is the faintest. All other variables in RoadRunner were fixed except for lanes.

After that, the vehicle was summoned by the RoadRunner Scenario function, and then driven on the same fixed route on the KIAPI map. While the car was driving in the center of the road, 25-second videos were recorded at 30 FPS on each map with specific characteristics of the lane. By extracting each frame of the recorded video, about 700 images were created for each map, and FID operation was performed between the maps using this. The FID code used in this paper was written based on PyTorch and the FID's dimensionality of features was set to 2048 (default) [27]. To compare only the thickness of lanes, we fixed the lane material to LaneMarking1, the length and spacing of lanes to (10m, 10m), and increased the lane thickness from 0.125m to 0.15m, 0.175m, and 0.2m respectively. Also, to compare the texture, the thickness was fixed to 0.125m and the length and spacing of the lanes were fixed to (10m, 10m). To compare the length and spacing of the lanes, the thickness was fixed to 0.125m and the material to LaneMarking1.

Table 5 shows the FID measurements between each map. The fact that the value increases toward the upper right of the table means that the FID actually appears larger as the characteristics of the lanes are quantitatively farther from each other. For example, the thickness changing result as shown in Table 5 (a) shows that the FID is smaller when the thickness of the lane is quantitatively close, and vice versa. When compared with the image of the lane with a thickness of 0.125 m, it can be seen that the FID increases from 3.36 (0.15 m) to 11.17 (0.2 m). For the texture of the lane as shown in Table 5(b), the image set with the clearest lane (LaneMarking1) was farther from the image set with the blurriest lane (LaneMarking3) than the distance from the image set with the second clearest lane (LaneMarking2). The same results as shown in Table 5(c) were obtained for the length and spacing of the lanes. This can be explained that the Inception v3 model abstracts the characteristics such as thickness, texture, length, and spacing of lanes well and extracts them as features. This tendency confirms that FID is suitable as a numerical value to quantitatively measure the difference in characteristics of the lanes.

B. EXTRACTING GROUND TRUTH

To automatically obtain the ground truth data in CARLA simulator, it was required to perform several works. As shown in Fig. 10, our works had following three steps. First, we created the same map except for the shape of the line. This map's line was fully connected shown as Fig. 10 (a). Then,

TABLE 5. FID measurement value between image sets with different (a) thicknesses of lanes, (b) materials of the texture of lanes, and (c) lane lengths & spaces between lanes.

Thickness	0.125m	0.15m	0.175m	0.2m
0.125m	0	3.36	7.57	11.17
0.15m		0	4.18	6.71
0.175m			0	3.12
0.2m				0

(a)

Texture	Lane Marking1	Lane Marking2	Lane Marking3
LaneMarking1	0	31.60	85.14
LaneMarking2		0	44.05
LaneMarking3			0

(b)

Length, Spacing	(10m, 10m)	(5m, 5m)	(3m, 3m)
(10m, 10m)	0	15.54	31.91
(5m, 5m)		0	11.09
(3m, 3m)			0

(c)

the vehicle ran in the line-connected map while recording its movement and collecting segmentation images. Second, the recording process was turned on in the original map, and then started to save the test images. Then, we found the specific pixel color belonging to the line and counted the number of points for each h_sample which is the height value that lanes can exist. Lastly, line order was designated.

This algorithm was implemented as described in Algorithm 1 only for the second road. When searching the lines based on height, the number of possible points in one height is 2, 3, or 4 (top red line of Fig. 10(b)). The two points mean line 2 and line 3 in order from the left (bottom red line of Fig. 10(b)). The three points mean line 2, line 3, and line 4 if a second point is on the right line based on the middle of the image (middle red line of Fig. 10(b)). Conversely, if the second point is on the left line, the three points mean line 1, line 2, and line 3. Lastly, the four points mean line 1, line 2, line 3, and line 4. Moreover, as shown in Fig. 11, this algorithm was successfully applied in the presence of diverse road conditions for the second lane (i.e., straight and curved lane).

C. COLLECTING GPS DATA

To compare the driving test results, we collected GPS data. The virtual vehicle was summoned to a map created

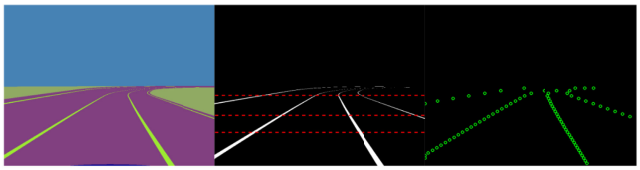


FIGURE 10. The segmentation image from line connected map (a), extracting the specific pixel color (b), and ground truth points (c). Red lines in (b) show the example of each case for Algorithm 1.

Algorithm 1 Automatically Extracting Ground Truth

Input Segmentation Images
Output Ground Truth Data

$h\_sample \leftarrow$ the list of height values that lines can exist
$line\_rgb \leftarrow$ the RGB value of segmented line
$w \leftarrow$ the width of images
$GT \leftarrow$ the 2D array of ground truth data. Row corresponds to h_sample and column corresponds to line number
for $i = h\_sample$ **do**
  for $j = 0, 1, 2, \ldots, w - 1$ **do**
    Append j in the $GT[i]$ that satisfies **Input**$[i][j] = line\_rgb$
  end for
  if $len(GT[i]) = 2$ **then**
    Set line 2, 3
  else
    if $len(GT[i]) = 3$ **then**
      if $GT[i][1] < w/2$ **then**
        Set line 1, 2, 3
      else
        Set line 2, 3, 4
      end if
    else
      Set line 1, 2, 3, 4
    end if
  end if
end for

FIGURE 11. Results of extracting ground truth (green points) on the original map images. (a) straight lane. (b) right curved lane. (c) left curved lane.

based on the GPS trajectory of the real vehicle and drove while performing LKAS. The starting position was (East, North) = (44549054.51, 394557162.27) as UTM. GPS data was collected through the following two experiments. One is trajectory comparison on a curved road, and the other is a

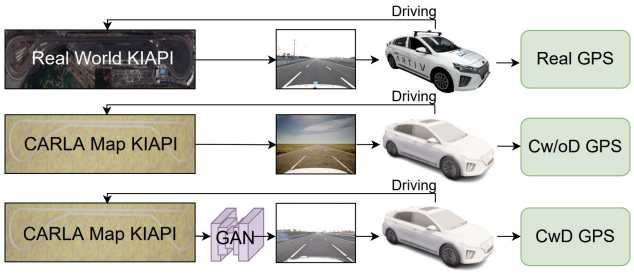


FIGURE 12. GPS setting and method flow for each domain.

comparison of lane restoring ability on a straight road. For trajectory comparison, the vehicle traveled 20 laps in each case of CwD and Cw/oD. For the lane restoring experiment, the vehicle was spawned at ± 90cm, ±120cm, ±130cm, ±140cm, and ±150cm in the direction perpendicular to the lane from the center. Fig. 12 shows how the proposed method is applied by using both test and simulation environments with respect to GPS measurement.

V. EXPERIMENTS AND RESULTS

To show the usefulness of DCLGAN in simulation driving tests, three methods were used to analyze data. First, FID thoroughly measured the reality of virtual-world images by calculating differences in the density of two distributions. FID showed DCLGAN made CARLA images to realistic. Second, lane segmentation performance, which was the important attribute for LKAS for autonomous vehicle, was measured and compared. These results show that ENet-SAD has a better understanding of images. Third, GPS data showed CwD images made driving results more realistic. We selected two road types: curved roads, and straight roads. Then, the curved road was analyzed by using Root Mean Square Error (RMSE) and the straight road was considered by the ability of lane restoring.

A. FID SCORE

Cw/oD2Real image gap was obtained by measuring the FID between 5498 CARLA images without DCLGAN and 6929 real-world images, and CwD2Real image gap was obtained by measuring the FID between 5298 images passed through DCLGAN and 6929 real-world images. Additionally, 4585 separate images captured independently were used for real to real comparison. This relationship is depicted in Fig. 13.

As a result of training by setting $\lambda_{NCE,X}$ to 3, the weight (epoch 26) obtained with FSIM value of 0.451 was used. An example of an image translated using this weight is shown in Fig. 14. The lanes of CARLA are inherently blurry and indistinguishable from the surrounding roads (Fig. 14 (a)), whereas the lanes of the real-world domain are clear and distinctly visible in the distance (Fig. 14 (b), (c)). This DCLGAN weight translates the image by well reflecting the difference in lane characteristics between these domains without distorting the shape of the lane.

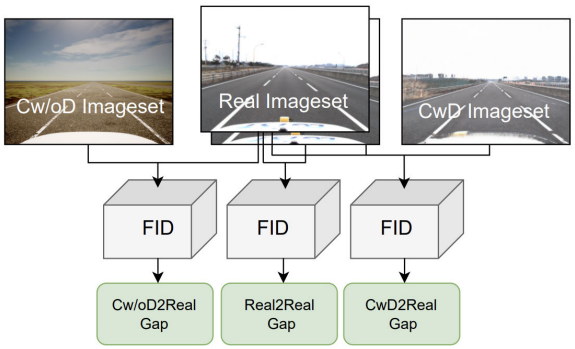


FIGURE 13. Cw/oD2Real gap (Cw/oD vs real-world image set), Real2Real gap (real-world image set vs another real-world image set), and CwD2Real gap (CwD vs real-world image set) measured by FID.

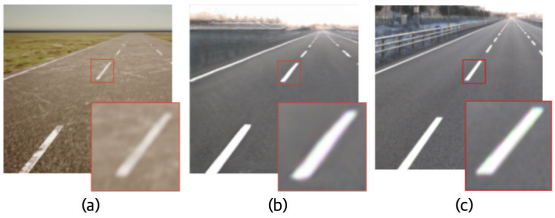


FIGURE 14. The lane on the real-world domain side is clearer, it can be easily distinguished from the surroundings, and the far side can be seen well. (a) Cw/oD, (b) paired CwD, and (c) real-world lane image for comparison.

TABLE 6. Real vs Cw/oD, Real vs CwD, and Real vs Real FID values for each dimensionality of the feature. The FID values of the CwD images subjected to DCLGAN were always smaller.

Feature Dimensionality	Real vs Cw/oD	Real vs CwD	Real vs Real
64	7.34	1.20	0.01
192	24.10	4.67	0.05
768	2.05	1.17	0.15
2048	321.97	240.80	86.50

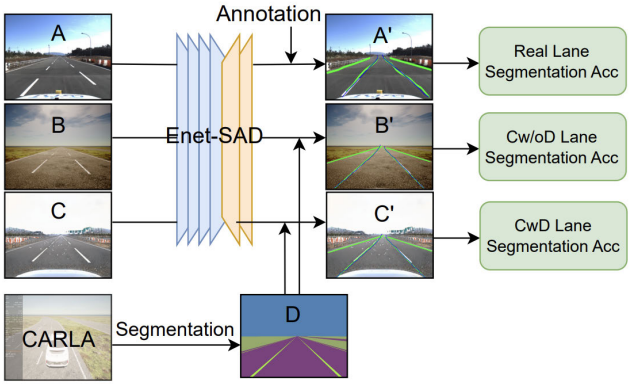


FIGURE 15. Specific flow for obtaining lane segmentation accuracy. A, B, C input image set for each domain. A′, B′, C′ green lines mean ground truth and blue lines mean ENet-SAD prediction results. D ground truth was obtained by the previously mentioned algorithm.

Table 6 is the FID measurement of the CARLA images before and after conversion using the corresponding weights.

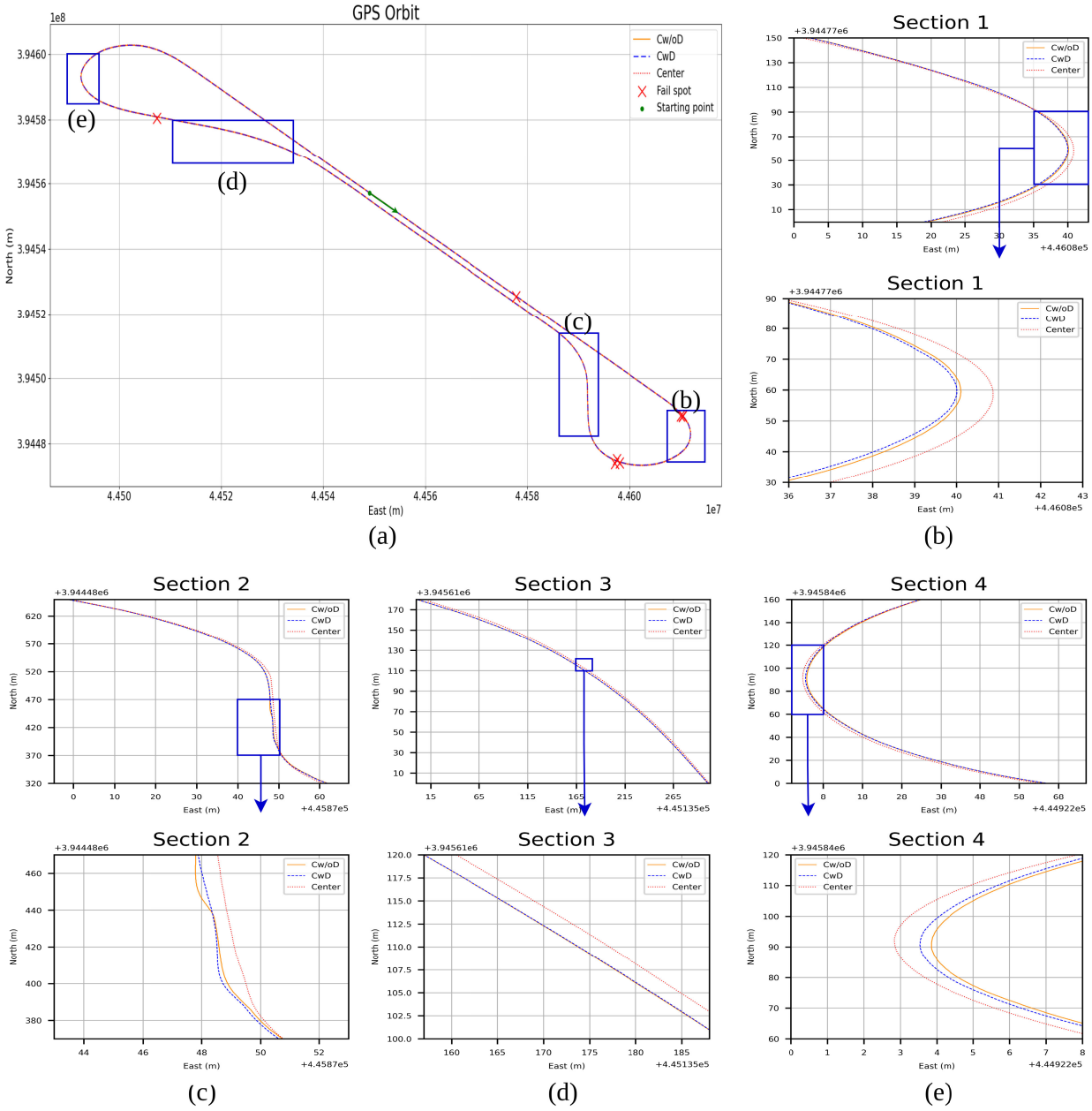


FIGURE 16. Example of GPS orbits of CwD, Cw/oD, and center path among 20 laps. (a) Position of each section. (b), (c), (d), (e) Magnified view of each section.

Considering the lack of diversity in our dataset, it was also measured when the dimensionality of the feature was 64, 192, and 768. In all cases, the FID value of CwD was lower than that of Cw/oD. (e.g. 321.97 (Cw/oD) → 240.80 (CwD) at 2048 feature dimensionality) That is, as we saw in the previous section, DCLGAN changed the CARLA image characteristics such as road color and lane clarity to fit the real-world domain, and it was possible to quantitatively confirm the reduction of the gap with FID.

B. LANE SEGMENTATION ACCURACY

Lane segmentation accuracy in each domain was evaluated and followed by the TuSimple evaluation method.

TABLE 7. Lane segmentation accuracy of Cw/oD, CwD and Real-world.

	Cw/oD	CwD	Real
Accuracy(%)	82.85	84.94	89.73

Prediction results were obtained using ENet-SAD used for LKAS mentioned in the previous section. As shown in Table 7, ENet-SAD shows better results in CwD(84.94%) than Cw/oD(82.85%). Fig. 15 describes specific flow.

TABLE 8. The degree of how much CwD and Cw/oD deviate from the center of the road in each session. CwD drove a complete lap in all cases, showing a 100% success rate. Cw/oD was 65.0%. Fail spot (X) in Fig. 16 shows the failed points during Cw/oD.

	Section1		Section 2		Section 3		Section 4		
	x (m)	y (m)	x (m)	y (m)	x (m)	y (m)	x (m)	y (m)	Success Rate (%)
CwD	0.627	2.082	0.233	1.571	0.178	0.107	0.531	1.003	100
Cw/oD	1.059	2.447	0.255	1.564	0.195	0.119	0.734	1.464	65.0
Cw/oD - CwD	0.433	0.365	0.022	-0.007	0.017	0.012	0.203	0.461	

This result shows that DCLGAN made CARLA images look like ENet-SAD training image dataset. As the training dataset was collected from real-world, CwD become similar to the real image. This made ENet-SAD easier to understand the images.

C. GPS ORBIT

1) DIFFERENCE OF GPS ORBIT

After data collection, to utilize RMSE, we selected two road types (i.e., slightly curved road, and steep curved road) and two sections for each road type. Fig. 16(a)-(d) described each section respectively. Then, the RMSE between the ideal path connecting the central points of the road and each driving path was calculated. More specifically, there are two ways to calculate RMSE: distance east (x) and distance north (y). Table 8 shows the 10% trimmed mean of RMSE values of CwD and Cw/oD for each section. The experiment was performed with the desired speed set at 50km/h turning on LKAS until finishing one lap. Then we did this experiment 20 times.

As shown in Table 8, CwD tended to run closer to the center route than Cw/oD for the most part. According to section I and section IV, the steep curved roads, CwD was more stable. However, in section II and section III which are slightly curved roads, CwD and Cw/oD were almost the same. Moreover, CwD finished whole laps but Cw/oD failed seven times. There are many failures on the bottom right corner road in Fig. 16 (a) which was the most curved road in the case of Cw/oD. These results of RMSE and success rates mean that Cw/oD can cause bad driving results, especially on the steep curved road which is the most important part of the map.

2) LANE RESTORING

In this section, we tried to evaluate how well a vehicle can return when it abruptly maneuvers out of its lane. If CwD recognizes the lane better than Cw/oD, the ability to return to the existing lane within reasonable time should be better when it deviates from the lane. In common, if the vehicle starts from its own location and returns to the original lane within 2 minutes and continues to keep the lane stably, it was judged as a success.

The results are shown in Fig. 17 and Table 9. As shown in Table 9, Cw/oD failed when starting by moving 1.5 m, 1.4 m to the right, and 1.2 m, 1.3 m, 1.4 m, and 1.5 m to the left. However, CwD succeeded in all but the right 1.5m, left 1.3m, 1.4m, and 1.5m. As shown in Fig. 17, the path returning to the original lane was confirmed even under relatively extreme

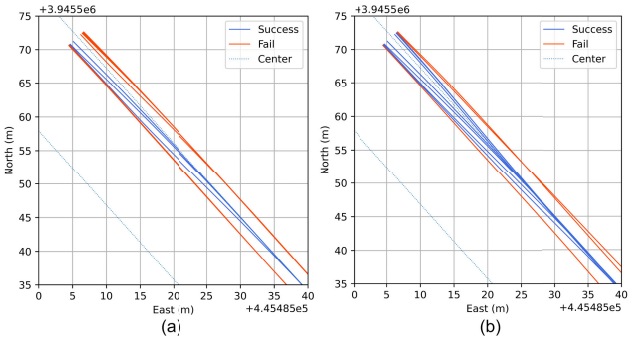


FIGURE 17. Lane restoring GPS path; Red (X), Blue (O). (a) result of Cw/oD (b) result of CwD.

TABLE 9. Lane return success or not according to the distance from the center of the lane (+ is right, − is left).

Distance (m)	1.5	1.4	1.3	1.2	0.9	-0.9	-1.2	-1.3	-1.4	-1.5
Cw/oD	X	X	O	O	O	X	X	X	X	X
CwD	X	O	O	O	O	O	O	O	X	X

conditions such as +1.4m and −1.3m unlike Cw/oD. Compared to Cw/oD, it can be seen that CwD has better restoring force to return to the center of the lane by better recognizing the lane even when it is above the lane away from the center of the lane.

VI. DISCUSSION AND CONCLUSION

Previous studies have shown ways to make virtual-world images close to the real-world images by improving GAN. However, these studies have not focused on the performance of GAN especially while driving test in simulation. In this paper, we evaluated the effect of translating simulation images into real image via DCLGAN on LKAS in CARLA simulation.

To measure the effect, we first found the well-trained DCLGAN weights while minimizing the distortion of the lanes using FSIM. Then FID quantitatively showed that virtual images can be more similar to reality through DCLGAN, especially in our KIAPI high-speed circulation environment. In addition, we made ENet-SAD better understand lane information of images by making the features of virtual images closer to the train set images of ENet-SAD by DCLGAN. In this process, we proposed an algorithm that automatically measures lane recognition accuracy in simulation. Finally, we compared the driving trajectory of critical parts through RMSE (x) and RMSE (y) and the ability of lane restoring.

These indicate that the driving result of CwD is more similar to reality. Thus, these results suggest that applying image translation on the simulation driving test can be essential and this visual gap can be reduced by using GAN.

Most notably, this is the first study to our knowledge to measure the performance of image translation on LKAS in autonomous simulation. Our results will make researchers in simulation field focus on image translation. Based on this study, our future work will concentrate on implementing scenarios for surrounding vehicles and testing autonomous driving algorithms such as LKAS and even AEB or SCC. In addition, a more accurate dynamic model will be applied through CARSIM with minimum gap of the simulation to real-world.

REFERENCES

[1] *The High Toll of Traffic Injuries: Unacceptable and Preventable*, World Bank, Washington, DC, USA, 2018. [Online]. Available: http://hdl.handle.net/10986/29129

[2] National Highway Traffic Safety Administration, "Standing general order on crash reporting for level 2 advanced driver assistance systems," Nat. Highway Traffic Saf. Admin., Washington, DC, USA, Tech. Rep. DOT-HS-813-325, 2022. [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-06/ADAS-L2-SGO-Report-June-2022.pdf

[3] R. Gutierrez, J. F. Arango, C. Gomez-Huelamo, L. M. Bergasa, R. Barea, and J. Araluce, "Validation method of a self-driving architecture for unexpected pedestrian scenario in CARLA simulator," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jul. 2021, pp. 1144–1149.

[4] D. J. Fremont, E. Kim, Y. V. Pant, S. A. Seshia, A. Acharya, X. Bruso, P. Wells, S. Lemke, Q. Lu, and S. Mehta, "Formal scenario-based testing of autonomous vehicles: From simulation to the real world," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–8.

[5] C. Gómez-Huélamo, J. Del Egido, L. M. Bergasa, R. Barea, E. López-Guillén, F. Arango, J. Araluce, and J. López, "Train here, drive there: Simulating real-world use cases with fully-autonomous driving architecture in CARLA simulator," in *Advances in Intelligent Systems and Computing*, 2020, pp. 44–59.

[6] Y. Chen, S. Chen, T. Xiao, S. Zhang, Q. Hou, and N. Zheng, "Mixed test environment-based vehicle-in-the-loop validation—A new testing approach for autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 1283–1289.

[7] C. Brogle, C. Zhang, K. L. Lim, and T. Braunl, "Hardware-in-the-Loop autonomous driving simulation without real-time constraints," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 3, pp. 375–384, Sep. 2019.

[8] S. Stević, M. Krunić, M. Dragojević, and N. Kaprocki, "Development of ADAS perception applications in ROS and 'software-in-the-loop' validation with CARLA simulator," *Telfor J.*, vol. 12, no. 1, pp. 40–45, 2020.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)* 2014, pp. 2672–2680.

[10] Y. Cao, Z. Zhou, W. Zhang, and Y. Yu, "Unsupervised diverse colorization via generative adversarial networks," 2017, *arXiv:1702.06674*.

[11] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," 2016, *arXiv:1609.04802*.

[12] E. Yurtsever, D. Yang, I. M. Koc, and K. A. Redmill, "Photorealism in driving simulations: Blending generative adversarial image synthesis with rendering," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 23114–23123, Dec. 2022.

[13] S. R. Richter, H. A. AlHaija, and V. Koltun, "Enhancing photorealism enhancement," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1700–1715, Feb. 2023.

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn. (CoRL)*, 2017, pp. 1–16.

[15] J. Han, M. Shoeiby, L. Petersson, and M. A. Armin, "Dual contrastive learning for unsupervised image-to-image translation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 746–755.

[16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232.

[17] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1–12.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[19] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection CNNs by self attention distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1013–1021.

[20] TuSimple. (2017). *Tusimple Benchmark*. GitHub repository. [Online]. Available: https://github.com/TuSimple/tusimple-benchmark

[21] H. Bae, G. Lee, J. Yang, G. Shin, G. Choi, and Y. Lim, "Estimation of the closest in-path vehicle by low-channel LiDAR and camera sensor fusion for autonomous vehicles," *Sensors*, vol. 21, no. 9, p. 3124, Apr. 2021.

[22] E. Seo, S. Lee, G. Shin, H. Yeo, Y. Lim, and G. Choi, "Hybrid tracker based optimal path tracking system of autonomous driving for complex road environments," *IEEE Access*, vol. 9, pp. 71763–71777, 2021.

[23] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial CNN for traffic scene understanding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 1–8.

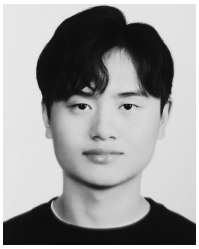
[24] KIAPI. *Proving Ground*. Accessed: Sep. 20, 2022. [Online]. Available: http://www.kiapi.or.kr/sub02/sub01_02.php?sid=2&page_num=&skey=&sval=&mode=

[25] CARLA Simulator. *Synchrony and Time-Step*. Accessed: Nov. 5, 2022. [Online]. Available: https://carla.readthedocs.io/en/latest/adv_synchrony_timestep/

[26] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: A feature similarity index for image quality assessment," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2378–2386, Aug. 2011, doi: 10.1109/TIP.2011.2109730.

[27] M. Seitzer, "Pytorch-FID," GitHub Repository, Tech. Rep., 2020. [Online]. Available: https://github.com/mseitzer/pytorch-fid

JINU PAHK was born in Daegu, South Korea, in March 2001. He is currently pursuing the B.S. degree in computer science and electrical engineering with the Daegu Gyeongbuk Institute of Science and Technology, Daegu. His research interests include deep learning and AI accelerator.



JUNGSEOK SHIM was born in Seoul, South Korea, in January 2001. He is currently pursuing the B.S. degree in computer science and technology with the Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, South Korea. His research interests include computer vision and computer networks.

MINHYEOK BAEK was born in Incheon, South Korea, in July 1999. He is currently pursuing the bachelor's degree with the College of Transdisciplinary Studies, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, South Korea. His research interests include virtual reality and human–computer interaction (HCI).

YONGSEOB LIM (Member, IEEE) received the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2010. He was with the Research and Development Center, Hyundai Motor Company, for about five years, from 2001 to 2006, as a Research Engineer. Moreover, he was also with the Mechatronics System Development Group, Samsung Techwin Company, South Korea, as a Principal Research Engineer, for about six years, from 2010 to 2016. His work with Samsung focused on developing stabilization and target tracking control algorithms for remotely controlled robotic systems. He is currently an Associate Professor with the Robotics and Mechatronics Engineering Department, Daegu Gyeongbuk Institute of Science and Technology (DGIST), and the Co-Director of the Joint Research Laboratory of Autonomous Systems and Control (ASC) and Vehicle in the Loop Simulation (VILS) Laboratories. His research interests include artificial intelligence (AI) algorithms of perception techniques, path planning, and tracking control for autonomous systems.

GYEUNGHO CHOI received the Ph.D. degree in mechanical engineering from The University of Alabama, Tuscaloosa, AL, USA, in 1992, and the Ph.D. degree (Hons.) in automotive and energy engineering conferred by His Royal Highness Princess Maha Chakri Sirindhorn from the King Mongkut's University of Technology North Bangkok, Thailand, in 2018. He has taught thermodynamics, heat transfer, and automotive engineering at Keimyung University and the Daegu Gyeongbuk Institute of Science and Technology (DGIST), as a Professor, for 30 years. He is currently the Co-Director of the Joint Research Laboratory of Autonomous Systems and Control (ASC) and Vehicle-in-the-Loop Simulation (VILS) Laboratories. His research interests include self-driving vehicles, ADAS systems, vehicle-in-the-loop systems, and the design of alternative energy utilization. He served as the President for the Korean Auto-Vehicle Safety Association and the Editor-in-Chief for the Korean Society for Engineering Education.